PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-171657
(43)Date of publication of application : 26.06.1998

(51)Int.Cl. G06F 9/44
G06F 9/46
G06F 13/00

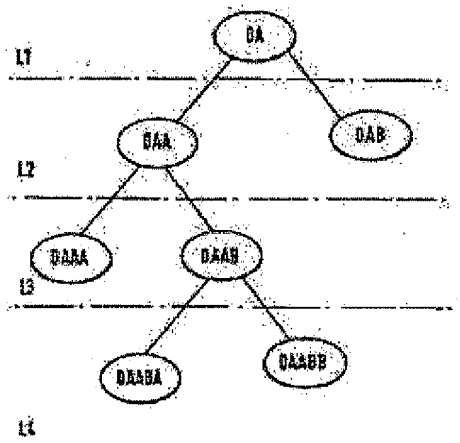
(21)Application number : 09-260776
(22)Date of filing : 20.08.1997

(71)Applicant : ALCATEL ALSTHOM CO GENERAL ELECTRICITE
(72)Inventor : CARRE LAURENT
POTONNIEE OLIVIER

(30)Priority
Priority number : 96 96440063 Priority date : 20.08.1996 Priority country : EP

(54) METHOD FOR SUPPORTING GENERATION OF FIRST OBJECT, METHOD FOR GENERATING FIRST OBJECT, METHOD FOR DELETING FIRST OBJECT, PROGRAM MODULE, AND COMPUTER UNIT

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a method for supporting the generation of a 1st object, a method for generating the 1st object, a method for deleting the 1st object, a program module, and a computer unit.
SOLUTION: In an object environment, the 1st object and another object are driven by using a common object request broker architecture(CORBA) mechanism. A 2nd object allowed to be used for a CORBA generation object and including at least one of functions for generating the 1st object is provided. A master to be an immediately preceding object in an including tree is allocated to the 1st object as a 2nd object. At least one function capable of generating the 1st object and corresponding to the CORBA generation object is integrated in the master object.



(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－171657

(43)公開日　平成10年(1998)６月26日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/44	５３０	Ｇ０６Ｆ	9/44	５３０Ｍ
	9/46	３６０		9/46	３６０Ｂ
	13/00	３５７		13/00	３５７Ｚ

審査請求　有　　請求項の数12　ＯＬ　外国語出願　（全 31 頁）

(21)出願番号　特願平9－260776

(22)出願日　平成９年(1997)８月20日

(31)優先権主張番号　９６４４００６３．４

(32)優先日　1996年８月20日

(33)優先権主張国　フランス（ＦＲ）

(71)出願人　391030332
アルカテル・アルストム・コンパニイ・ジエネラル・デレクトリシテ
ＡＬＣＡＴＥＬ　ＡＬＳＴＨＯＭ　ＣＯＭＰＡＧＮＩＥ　ＧＥＮＥＲＡＬＥ　Ｄ'ＥＬＥＣＴＲＩＣＩＴＥ
フランス国、75008　パリ、リュ・ラ・ボエテイ　54

(72)発明者　ロラン・カレ
フランス国、エフー78960・ボワザンールーブルトヌー、リユ・バン・ゴツグ、26

(74)代理人　弁理士　川口　義雄　（外１名）

最終頁に続く

(54)【発明の名称】　第１のオブジェクトの生成をサポートするための方法、第１のオブジェクトを生成するための方法、第１のオブジェクトの削除のための方法、プログラムモジュール、およびコンピュータユニ

(57)【要約】

【課題】　第１のオブジェクトの生成をサポートするための方法、第１のオブジェクトを生成するための方法、第１のオブジェクトの削除のための方法、プログラムモジュール、およびコンピュータユニットを提供する。

【解決手段】　オブジェクト環境において第１のオブジェクトと他のオブジェクトがＣＯＲＢＡ機構を使用して動作する。ＣＯＲＢＡ生成オブジェクトであり、第１のオブジェクトの生成のための少なくとも一つの機能を含む第２のオブジェクトを提供する。次に、包含ツリー内で直前のオブジェクトである親オブジェクトを、第２のオブジェクトとして第１のオブジェクトに割り当てる。さらに、第１のオブジェクトを生成し、ＣＯＲＢＡ生成オブジェクトに対応する少なくとも一つの機能を、この親オブジェクトに組み込む。

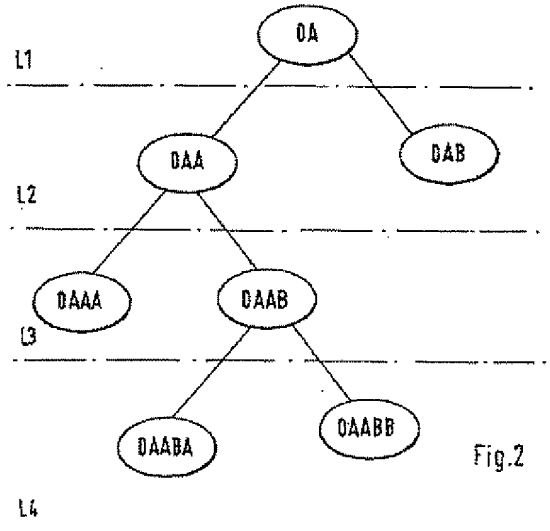


Fig.2

1

【特許請求の範囲】

【請求項１】　第１のオブジェクトと他のオブジェクトがＣＯＲＢＡ機構を介してインタラクトするオブジェクト環境において第１のオブジェクトの生成をサポートする方法であって、該方法を使用して、第１のオブジェクトの生成のための少なくとも一つの機能を含むＣＯＲＢＡ生成オブジェクトである第２のオブジェクトが提供されるようになっており、第１のオブジェクトが包含ツリーに従って直前にある親オブジェクトに第２のオブジェクトとして割り当てられ、ＣＯＲＢＡ生成オブジェクトに対応する第１のオブジェクトの生成のための少なくとも一つの機能が該親オブジェクトに組み込まれることを特徴とする方法。

【請求項２】　割当てが中央サービスに登録されることを特徴とする請求項１に記載の方法。

【請求項３】　第１のオブジェクトの削除のための機能に対応する少なくとも一つのＣＯＲＢＡ生成オブジェクトが親オブジェクトに組み込まれることを特徴とする請求項１に記載の方法。

【請求項４】　ＣＭＩＳＥ機能が親オブジェクトに生成機能として組み込まれることを特徴とする請求項１に記載の方法。

【請求項５】　親オブジェクトの生成機能がＣＯＲＢＡ子オブジェクトに渡されることを特徴とする請求項１に記載の方法。

【請求項６】　第１のオブジェクトと他のオブジェクトがＣＯＲＢＡ機構を介して対話するオブジェクト環境内において第１のオブジェクトの生成を行う方法であって、該方法の過程でＣＯＲＢＡ生成オブジェクトに生成メッセージが送られるようになっており、第１のオブジェクトに対する生成メッセージが包含ツリーに従って直前にある親オブジェクトに送られ、ＣＯＲＢＡ生成オブジェクトとして機能する親オブジェクトが第１のオブジェクトの生成を行うことを特徴とする方法。

【請求項７】　第１のオブジェクトと他のオブジェクトがＣＯＲＢＡ機構を使用して対話するオブジェクト環境内において第１のオブジェクトの削除を行う方法であって、該方法の結果としてＣＯＲＢＡ生成オブジェクトに削除メッセージが送られるようになっており、第１のオブジェクトに対する削除メッセージが包含ツリーに従って直前にある親オブジェクトに送られ、ＣＯＲＢＡ生成オブジェクトとして機能する親オブジェクトが第１のオブジェクトの削除を行うことを特徴とする方法。

【請求項８】　ＣＯＲＢＡ機構を使用してＣＯＲＢＡオブジェクトとしてインタラクトするためのＣＯＲＢＡインタフェースと、アプリケーションサービスの確立のための一組の一次機能とを備えるプログラムモジュールであって、ＣＯＲＢＡ生成オブジェクトとしてサービスを提供する形態を有し、包含ツリーに関する論理的観点から見て直後にある子オブジェクトである子オブジェクトの生成をプログラムモジュールから行う、一つまたは複数の二次機能を含むことを特徴とするプログラムモジュール。

2

【請求項９】　第１の機能がネットワーク管理のためのアプリケーションサービスであることを特徴とする請求項８に記載のプログラムモジュール。

【請求項１０】　第２の機能のセマンティックがＣＯＲＢＡ生成オブジェクトであることを特徴とする請求項８に記載のプログラムモジュール。

【請求項１１】　請求項８に記載のプログラムモジュール上で動作するコンピュータユニット。

【請求項１２】　コンピュータユニットがネットワーク管理構成要素であることを特徴とする請求項１１に記載のコンピュータユニット。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、請求項１の包括項による第１のオブジェクトとその他のオブジェクトがＣＯＲＢＡ機構を使用してインタラクトするオブジェクト環境内の第１のオブジェクトの生成をサポートする方法と、請求項６または７の包括項による第１のオブジェクトの生成と削除を行う方法と、請求項８の包括項によるＣＯＲＢＡ機構を使用してＣＯＲＢＡオブジェクトとしてインタラクトするためのＣＯＲＢＡインタフェースを備えたプログラムモジュールと、請求項１１の包括項によるコンピュータユニットとに関する。

【０００２】

【従来の技術】分散コンピュータシステムと共に使用するソフトウェアの設計のためのアーキテクチャ原理としてオブジェクト指向モデリングがますます使用されるようになっている。ＣＯＲＢＡソフトウェア（ＣＯＲＢＡ＝コモンオブジェクトリクエストブローカアーキテクチャ）は、そのようなアーキテクチャを備え、オブジェクトマネジメントグループ（ＯＭＧ）によって規定されたＯＳＡアーキテクチャ（ＯＳＡ＝オブジェクトーサービスアーキテクチャ）の重要な構成要素である。

【０００３】本発明は、ＣＯＲＢＡアーキテクチャによるコンピュータシステムにおいてオブジェクト（管理オブジェクト）の生成と削除を通常行うための方法に基づく。これについては、たとえば「Ｃｏｍｍｏｎ　Ｏｂｊｅｃｔ　Ｒｅｑｕｅｓｔ　Ｂｒｏｋｅｒ：Ａｒｃｈｉｔｅｃｔｕｒｅ　ａｎｄ　Ｓｐｅｃｉｆｉｃａｔｉｏｎｒ　２．ｏ」（オブジェクトマネジメントグループ、米国マサチューセッツ州フレーミングハム、１９９５年）に記載されている。

【０００４】オブジェクトの生成と削除は、特殊なオブジェクト、すなわち生成オブジェクト（ファクトリ）を使用して行われる。このような生成オブジェクトの唯一の機能は、特定の種類のオブジェクトについてオブジェクトの削除と生成を行うことである。特殊なサービスで

あるライフサイクルサービスを使用して、このような生成オブジェクトを定義する。汎用生成オブジェクトを使用することができる。この結果として、特殊な生成パラメータの入力を可能にする、たとえば特殊なクラスのオブジェクト（オブジェクトクラス）などのあらゆる種類のオブジェクトの特殊な生成オブジェクトを定義することができる。

【０００５】一つのみの特殊オブジェクトの生成または消去を行うことができるためには、まず第１に、その特殊オブジェクトの生成と削除を行う適切な生成オブジェクトを見つけなければならない。これを行うには、ライフサイクルサービスの一部である特殊サービスのファクトリファインダサービスに探索メッセージを送り、その探索のための探索基準（タイプ、場所など）を入力する。適切な生成オブジェクトが見つかった場合、要件メッセージ（呼出し）が生成され、その生成オブジェクトが送られ、生成オブジェクトがその呼出しに含まれているパラメータに対応するオブジェクトの生成または消去を行う。

【０００６】この方法の欠点は、通常は適切な生成オブジェクトの探索が時間のかかる活動であり、その結果としてコンピュータの通信負荷がさらに増大することである。

【０００７】

【発明が解決しようとする課題】本発明の基本課題は、ＣＯＲＢＡアーキテクチャを備えたコンピュータシステムにおけるシステムの負荷を減らすことである。

【０００８】

【課題を解決するための手段】この課題は、請求項１、６、および７に記載の原理による方法と、請求項８に記載の原理によるプログラムモジュールと、請求項１１の原理によるコンピュータユニットとによって解決される。

【０００９】なお、本発明は、ＣＯＲＢＡ環境におけるオブジェクトが、包含ツリー内でそのオブジェクトの直後にあるオブジェクト（子孫オブジェクト）の生成または削除を行うという概念に基づいている。さらに、したがってそれらのオブジェクトはその子オブジェクトの生成オブジェクトの役割を果たす。

【００１０】ＣＯＲＢＡの原理とは異なるこの方法の結果として、以下のような利点が得られる。

【００１１】コンピュータシステムにおけるオブジェクトの数が少なくなり、その結果、システム負荷と参照数も少なくなる。

【００１２】さらに、ファクトリファインダサービスの必要がなくなり、探索手続きによるコンピュータ負荷が不要になる。包含ツリー内での配列のために、オブジェクトが容易に見つけられる。

【００１３】さらに、システム整合性が向上する。後続オブジェクトは、包含ツリー内にその先行オブジェクトが存在する場合にのみ生成される。

【００１４】ＣＯＲＢＡに基づくネットワーク管理システムが存在するネットワーク管理の一般分野におけるハイブリッドシステムの結果として、さらに他の利点が得られる。たとえば、すべてのＣＭＩＳＥサービスが保持されている単純なインタフェースから継承されるＣＭＩＳＥサービスを、生成機能および消去機能に使用することができる。

【００１５】

【発明の実施の形態】以下では、本発明について、添付図面によって補足される三つの詳細な例を示して詳述する。

【００１６】第１の実施形態では、本発明による一つまたは複数のコンピュータユニットから成り、各コンピュータユニット内で本発明によるプログラムモジュールが稼働している、本発明によるコンピュータシステムにおける方法の実行について説明する。

【００１７】図１に、互いに通信するコンピュータユニットＣ１～Ｃ３を有するＣＳコンピュータシステムを示す。

【００１８】典型的には、コンピュータユニットＣ１～Ｃ３は、通信ネットワークにおけるコンピュータ、プリンタ、またはネットワーク要素から成る。各コンピュータユニットは、プロセッサ、メモリ機構、および周辺構成装置から成るハードウェアプラットフォームと、たとえばオペレーティングシステムやデータバンクシステムなどのソフトウェアプラットフォームと、ソフトウェアプラットフォームから稼働するアプリケーションプログラムモジュールから成るアプリケーションとを有する。コンピュータユニットＣ１～Ｃ３は、たとえばＸ．２５、＃７、イーサネット、またはトークンリング通信システムなどの一つまたは複数の通信ネットワークによって互いに接続されている。コンピュータユニットＣ１～Ｃ３のソフトウェアプラットフォームは、これによって必要なデータ伝送サービスを提供する。

【００１９】アプリケーションプログラムモジュールは、オブジェクト（管理オブジェクト）としてモデル化される。すなわちオブジェクトのコードおよびデータは、他のオブジェクトがアクセスすることができる属性と機能を合わせたものによって表される。コンピュータシステムＣＳのアプリケーション機能は、多数のそのようなオブジェクト間の相互アクセスの結果として実現される。

【００２０】ＣＯＲＢＡアーキテクチャによると、コンピュータユニットＣ１～Ｃ３はいくつかのオブジェクトＣＯおよびＳＯといくつかのオブジェクト要求ブローカＯＲＢを提示する。

【００２１】サービスの点から見ると、オブジェクトＣＯおよびＳＯはそれぞれ、クライアントが必要とする可能性がある一つまたは複数のサービスを提供するカプセ

ル化された単位とみなすことができる。このようにして、オブジェクトＣＯ（クライアントオブジェクト）はサービスを要求し、そのサービスはＳＯオブジェクト（サーバオブジェクト）によって提供される。

【００２２】サービスを要求するために、ＣＯはＳＯに要求を送る。このような要求には、一つの操作と、一つの目的オブジェクトと、ゼロまたはいくつかのパラメータと、任意選択により要求のコンテキストの情報が含まれる。サービスが提供された後、ＳＯはその要求メッセージのために指定された結果メッセージをＣＯに返す。

【００２３】要求メッセージと結果メッセージを送受信することができるように、ＳＯオブジェクトとＣＯオブジェクトはインタフェースユニットＩＵを使用することができる。

【００２４】オブジェクト要求ブローカ（ＯＲＢ）は、分散環境でオブジェクトが通信することができるようにする基盤を提供する。なお、サービスの提供を要求されるＳＯの場所はオブジェクトＣＯにとって重要ではない。すなわち、他のコンピュータユニットのうちのどのコンピュータユニットにあるかは重要ではなく、オブジェクトがどの特別なプラットフォームまたはインプリメント方法で実現されたかも重要ではない。

【００２５】なお、各オブジェクトは少なくとも一つのオブジェクト要求ブローカ（ＯＲＢ）を知っており、このローカルオブジェクト要求ブローカとの接触方法も知っている。各オブジェクト要求ブローカは、他のオブジェクト要求ブローカとの接触方法およびそれらのオブジェクトとの通信方法を知っている。この状況では、オブジェクト要求ブローカはＲＰＣ手続き（ＲＰＣ＝遠隔手続き呼出し機構）を使用する。これによって、オブジェクトは要求メッセージを送り、オブジェクト要求ブローカＯＲＢの一つが、オブジェクト要求ブローカによって構築されたＣＯＲＢＡ基盤を使用してそのメッセージが目的オブジェクトに確実に渡されるようにする。

【００２６】図１ｂに、ＣＯとＳＯとの間の通信のための通信機構の図を示す。図１ｂには、通信層ＯＲＢコアと、その上をおおう五個の機能ユニットＤＩＩ、ＩＤＬＳｕｂｓ、ＯＲＢＩ、ＳＫＥＬ、およびＢＯＡを含む通信層が、これらの機能ユニットにアクセスすることができる二つのオブジェクトＣＯおよびＳＯと共に図示されている。

【００２７】ＣＯＲＢＡ基盤を介して通信し、この基盤内の他のオブジェクトと協調動作することができるためには、オブジェクトＣＯとＳＯはそれぞれＣＯＲＢＡ固有のインタフェースにアクセスすることができなければならない。なお、このようなインタフェースには、他のオブジェクトが当該オブジェクトに要求することができる可能な操作のセットの記述が含まれている。この場合、オブジェクトインタフェースは純粋にインタフェース記述言語のみから成る記述言語ＩＤＬ（インタフェース定義言語）で定義される。このインタフェースの継承によって、一つのオブジェクトがいくつかのインタフェースをサポートすることが可能になる。

【００２８】ＣＯＲＢＡでは、このＣＯＲＢＡ固有インタフェースを介してオブジェクトに直接アクセスが行われる。このインタフェースのインプリメンテーションはオブジェクト自体である。これは、コードとデータから成り、オブジェクトが単にデータ構造体のみによって表現される場合のようなエージェントエンティティを必要としない。

【００２９】要求メッセージを送るために、オブジェクトＣＯはオブジェクトＳＯのレファレンス（オブジェクトレファレンス）にアクセスし、オブジェクトＳＯのタイプと実行する操作を知る必要がある。オブジェクトＣＯは、ＩＤＬＳｔｕｂｅ機能ユニットのサブルーチンを呼び出すことによって、または機能ユニットＤＩＩ（動的呼出しインタフェース）を使用して動的にメッセージを要求することによって、要求メッセージを開始する。この二番目の手続きを使用すると、オブジェクトＣＯが作成された時点でまだ未知だったサービスの要求を行うことができる。

【００３０】オブジェクトＳＯでの要求メッセージの受信は、機能ユニットＤＯＡ（基本オブジェクトアダプタ）を使用してサポートされる。オブジェクトは、前述の第２の場合に対応する機能ユニットＳＫＥＬの機能を使用してインタフェースを提供することもできる。

【００３１】コンピュータシステムＣＳのオブジェクト間には論理関係があり、その構造を図２に例示する。

【００３２】図２には、七個のオブジェクトＯＡないしＯＡＡＢＢが図示されており、それらの間には関係がある。このオブジェクト相互の関係は、包含ツリーまたは包含階層と呼ばれる。各オブジェクトは特殊オブジェクトクラスの例（オブジェクトインスタンス）を表す。オブジェクトクラスは特殊階層を有し、したがって、たとえば、より特化されたオブジェクトクラスがより汎化されたクラスに包含され、より汎化されたオブジェクトクラスはいくつかのより特化されたオブジェクトクラスを包含する。これと同じことは具体的なオブジェクト（管理オブジェクト、オブジェクトインスタンス）にも当てはまる。オブジェクトＯＡは包含ツリーの根を形成する。オブジェクトＯＡＡおよびＯＡＢは、より上位のオブジェクトＯＡに包含される、より特化されたオブジェクトである。オブジェクトＯＡＡＡおよびＯＡＡＢは、オブジェクトＯＡＡ内に包含されるオブジェクトであり、オブジェクトＯＡＡＢＡおよびＯＡＡＢＢはオブジェクトＯＡＡＢに包含されるオブジェクトである。より上位のオブジェクトは、包含されるオブジェクトの親オブジェクトとも言われ、包含されるオブジェクトはその子オブジェクトと言うことができる。

【００３３】ＣＯＲＢＡでは、特殊な生成オブジェクト

（ファクトリ）を使用するオブジェクトの生成と削除を、サービスの要求および提供とまったく同じようにして行うことができる。これらの各生成オブジェクトは、どの時点でのどの特殊オブジェクトの生成と削除にも適合した操作のセットを含む。

【００３４】この操作のセットは次に、特殊生成オブジェクトからその主なタスクが実際には全く異なるサービスの提供である他のオブジェクトに転置される。これらの操作の転置は以下の方式に従って進められる。包含ツリーに従って割り当てられた子オブジェクトの生成と削除を行う各親オブジェクトにこの操作のセットが転置される。子オブジェクトの生成によって、親オブジェクトにこの操作のセットが継承され、生成プロセスにおいてさらに特化することができ、その結果、たとえばより多くのパラメータを持つより特化された生成機能が可能になる。したがって、オブジェクト（管理オブジェクト）は、依存関係ツリー内ですぐ後に続くオブジェクトの生成オブジェクト（ファクトリ）を形成する。オブジェクトのこれらの生成活動と削除活動を管理するために、オブジェクトにはそのすぐ後に続く可能性のあるオブジェクトのタイプのリストが格納される。

【００３５】このようにして、このオブジェクトのインタフェースは、元々あった操作に加えて、包含ツリー内で直後に続くオブジェクトの生成機能および削除機能も含む。

【００３６】このようにしてオブジェクトを生成または削除する場合、対応する要求メッセージが適切な親オブジェクトに送られる。この要求メッセージのセマンティクスは、生成オブジェクトのセマンティクスである。論理関係のため親オブジェクトを容易に突き止めることができ、その結果、責任を負う生成オブジェクトを簡単に見つけ出すことができる。

【００３７】この形態のオブジェクトの生成と消滅をコンピュータシステムＣＳのすべてのオブジェクトに適用する必要はなく、包含ツリーの枝に割り当てられたオブジェクトのみに適用すればよいということも可能である。削除機能を生成機能から分離することさえも可能である。したがって、削除操作を、削除するオブジェクトのインタフェースに組み込むこともできる。

【００３８】まず特殊クラスのオブジェクトの生成オブジェクトを指定する必要がある場合、すでに存在する既存機能にさかのぼることもできる。

【００３９】コンピュータシステムでＣＭＩＳＥ操作を使用可能な場合、生成機能と削除機能にＣＭＩＳＥ操作を使用することができ、さらにＣＭＩＳＥインタフェースで継承することができる。このようなさらに継承可能なインタフェースの該当部分を、記述言語で図３に具体的に示す。

【００４０】なお、図３の操作は以下のような意味を持つ。

【００４１】－Ｓｕｐｐｏｒｔ＿ｓｕｂｏｒｄｉｎａｔｅは、この関係ではサポート操作という汎用ファクトリの指定である。より汎用的なキーではなくオブジェクト識別子を入力値として持つ。

【００４２】－Ｃｒｅａｔｅ＿ｓｕｂｏｒｄｉｎａｔｅは、ｃｒｅａｔｅ＿ｏｂｊｅｃｔという汎用ファクトリを特化したものである。この場合もより汎用的なキーではなくオブジェクト識別子を使用する。「ｎａｍｅ」という特別なパラメータがあり、これは生成されたオブジェクトの属性を使用してそのオブジェクトに名前を付ける。しかし、「ｎａｍｅ」パラメータは最後の「ｃｒｉｔｅｒｉａ」パラメータに含めることもできる。これは、ｃｒｅａｔｅ＿ｏｂｊｅｃｔという汎用ファクトリの場合と同じである。＜ｎａｍｅ，ｖａｌｕｅ＞の対のリストがあり、これには生成パラメータ（初期値、資源制約、場所、．．．．．）のいずれか一つを入れることができる。レファレンスオブジェクトがある場合、レファレンスオブジェクトは基準の中に格納される。例外は、汎用ファクトリｃｒｅａｔｅ＿ｏｂｊｅｃｔによって報告し戻される。ｎａｍｅパラメータが無視されたことを示すａｕｔｏｎａｍｉｎｇという戻り報告の場合を除き、オブジェクトは自動的に命名される。

【００４３】－Ｄｅｌｅｔｅ＿ｓｕｂｏｒｄｉｎａｔｅは、ｒｅｍｏｖｅというライフサイクルオブジェクトを特化したものである。この要件は、オブジェクトの削除後に解放される資源を管理することができるように親オブジェクトに宛てて送られる。この操作自体が、逆転されたオブジェクト自体における逆転操作を出すことができ、それによって、オブジェクトが逆転可能であるかどうかを検査することができ、それによってそれ自体の資源を消去することができる。Ｄｅｌｅｔｅ＿ｓｕｂｏｒｄｉｎａｔｅは、ｒｅｍｏｖｅライフサイクルオブジェクトを使用するのと全く同様に、「除去不能バリエーション」を無効にすることができる。これは、提供された名前がどの既存の子オブジェクトも識別しないということを表す追加の無効名バリエーションを持つ。

【００４４】第２の実施形態によって、ネットワーク管理領域の状況を示す。

【００４５】第２の実施形態では、一つまたは複数の本発明によるコンピュータユニットから成り、各コンピュータユニット内で本発明によるプログラムモジュールが稼働している、本発明によるコンピュータユニットにおける本発明による方法の実行について説明する。第１の実施形態とは異なり、コンピュータシステムに関するこの態様は、非ＣＯＲＢＡオブジェクトが、外部から見るとＣＯＲＢＡ基盤上でＣＯＲＢＡオブジェクトであるかのように動作するように、適応化手続きによって修正されるネットワーク管理システムに関する。このコンピュータシステムは図１ａのコンピュータシステムＣＳと同様に装備されている。コンピュータユニットは、たとえ

ばネットワーク要素、ネットワーク管理センタ、または調停装置などのネットワーク管理ユニットを表す。コンピュータシステムはさらにネットワークサービスの管理や提供などのタスクも行うことができる。これらのサービスはＴＩＮＡソフトウェアアーキテクチャ（１９９４年版ＴＩＮＡ　ＤＰＥサービス仕様ＴＩＮＡ－Ｃ）に基づくこともできる。

【００４６】図４ａは、ＣＯＲＢＡ基盤を介した二つのこのようなネットワーク管理オブジェクト間の通信のための通信機構を示す図である。

【００４７】図４ａには、通信層ＣＯＲＢ／ＯＲＢと、この通信層を使用して一般に使用可能ないくつかのＣＭＩＳＥサービスと、二つのネットワーク管理構成要素ＭおよびＡと、任意の時点において後者のオブジェクトと通信層ＣＯＲＢ／ＯＲＢの間にある二つの通信機能ＧＭＯ／Ｃ＋＋およびＣＭＩＳＥ／ＩＤＬとが示されている。

【００４８】同様に、ネットワーク管理の分野のためにＯＳＩ（開放型システム間相互接続）によってオブジェクトモデルが標準化されている（開放型システム間相互接続のための管理フレームワーク、１９９２年版ＩＴＵ－Ｔ勧告Ｘ．７００）。このオブジェクトモデル（ＳＭＩ＝管理情報の構造）に加えて、オブジェクト間の通信のために、基本オブジェクトである一組の管理サービス（ＣＭＩＰ＝共通管理情報プロトコル）も規定されている。オブジェクトは記述言語ＧＥＭＯで指定され、この記述言語はＡＳＮ構文を使用し、それ独自の追加のマクロを含む。

【００４９】構成要素ＭおよびＡの場合、これらはＣＯＲＢＡオブジェクトではなく、一つまたは複数のＯＳＩオブジェクトＯＭまたはＯＡであり、マネージャまたはエージェント機能ユニットである。マネージャまたはエージェント機能ユニットを使用して、オブジェクトに対する操作が行われたり、要求メッセージが他のオブジェクトに送られたりする。

【００５０】エージェントおよびマネージャ機能ユニットは、ＣＭＩＰプロトコルを介して通信する。ネットワーク管理の観点から見ると、構成要素Ｍはマネージャの役割を果たし、構成要素Ａはエージェントの役割を果たす。

【００５１】通信ユニットＧＤＭＯ／Ｃ＋＋は、オブジェクトＯＡまたはオブジェクトＯＭに対するＣＭＩＳＥ操作の実行を可能にする一つまたは複数の特殊なアクセスオブジェクトから成る。

【００５２】ＣＭＩＳＥ管理サービスは、オブジェクトＯＡ側のＣＭＩＳＥオブジェクトによって実現される。インタフェースユニットＣＭＩＳＥ／ＩＤＬは、このＣＭＩＳＥオブジェクトとこのオブジェクトに割り当てられたサービスとを含む。インタフェースＣＭＩＳＥ／ＩＤＬのＣＭＩＳＥオブジェクトは、ＩＤＬインタフェースオブジェクトによって指定され、外部的にはＣＯＲＢＡオブジェクトのように動作し、ＣＯＲＢＡオブジェクトのように見える。この指定を可能にし、それによってオブジェクトＯＡにＣＯＲＢＡインタフェースを提供するために、ＡＳＮ．１タイプからＩＤＬタイプへのタイプ変換が必要である。このようにして、ＣＭＩＳＥサービスを一組のＣＯＲＢＡオブジェクトとして使用することができる。したがって、ＣＯＲＢＡ基盤を介して供給されるＣＯＲＢＡ要求を使用してオブジェクトＯＡに対してＣＭＩＳＥ操作を行うことができる。これと同じことはオブジェクトＭＯにも当てはまる。

【００５３】インタフェース内の図３に示されている部分はさらに継承されて、オブジェクトの生成または削除を可能にする。

【００５４】ＣＯＲＢＡ基盤を介してＯＳＩオブジェクトを結合する第２の可能性を、第３の実施例を表す図４ｂに示す。

【００５５】図４ｂには、通信層ＣＯＲＢ／ＯＲＢと、この通信層を使用して一般に使用可能ないくつかのＣＭＩＳＥサービスと、オブジェクトＯＭおよびＯＡと、任意の時点において後者のオブジェクトと通信層ＣＯＲＢ／ＯＲＢとの間にある通信機能ＧＤＭＯ／ＩＤＬおよびＣＭＩＳＥ／ＩＤＬとが図示されている。

【００５６】インタフェースユニットＧＤＭＯ／ＩＤＬを使用して、構成要素ＡおよびＭの指定されたＯＳＩオブジェクトがＩＤＬインタフェースとして仕様に変換される。このような指定されたオブジェクトへのアクセスは、従来のＣＯＲＢＡメッセージを使用して行うことができる。このようにして、各ＯＳＩオブジェクトが純粋なＣＯＲＢＡオブジェクトに変換される。ＩＤＬとＡＳＮ．１の仕様は性質が異なるため（インタフェース記述<ー>オブジェクト仕様）、完全な変換は不可能であり、ＣＭＩＳＥサービスのサブセットのみがインタフェースユニットＧＤＭＯ／ＩＤＬを介して提供される。これは、変換されたＣＯＲＢＡオブジェクトに対してＣＭＩＳＥ操作のサブセットしか実行できないことを意味する。

【００５７】ＣＭＩＳＥサービスを使用したオブジェクトの削除と生成を容易にするため、追加のインタフェースユニットＣＭＩＳＥ／ＩＤＬを提供する。ＣＭＩＳＥに存在する生成機能（m生成）にはＣＯＲＢＡインタフェースを使用してアクセスすることができるようになり、したがって変換されたＣＯＲＢＡオブジェクトに対するこの機能の適用が可能になる。

【００５８】構成要素Ａにおいてオブジェクトを生成するには、オブジェクトの生成および削除を行う生成オブジェクトの管理のために、構成要素Ｍのオブジェクトまたはコンピュータシステム ＣＳの特殊なＣＯＲＢＡオブジェクトをアドレスしなければならない。この生成オブジェクトは、生成するオブジェクトの親オブジェクトで

あり、これは同様に構成要素Ａのオブジェクトである。生成機能へのアクセスを獲得することができるためには、ここでＣＭＩＳＥ／ＩＤＬインタフェースユニットを介してＣＭＩＳＥ生成機能へのアクセスを行わなければならず、これによってこのインタフェースを介して親オブジェクトにアクセスすることができるようになる。これによってこの子オブジェクトの生成は、ＣＭＩＳＥセマンティクスを利用するＣＯＲＢＡメッセージを使用し、変換済みＣＯＲＢＡオブジェクトを使用してＣＭＩＳＥサービスのサポートにより行われる。逆の方式で、対応するＣＭＩＳＥサービスへのアクセスを行ってオブジェクトを消去する。

【図面の簡単な説明】

【図１ａ】第１の実施例のコンピュータシステムを示す＊ブロック回路図である。

【図１ｂ】図１に示すコンピュータシステムのソフトウェアの構造を示す機能図である。

【図２】オブジェクト間の依存関係を示す記号図である。

【図３】記号記述段階における表現のインタフェース記述から抽出した部分を示す図である。

【図４ａ】第２の実施例のコンピュータシステムのソフトウェア構造を示す機能図である。

【図４ｂ】第３の実施例のコンピュータシステムのソフトウェア構造を示す機能図である。

【符号の説明】

Ｃ１、Ｃ２、Ｃ３　コンピュータユニット

【図１ａ】

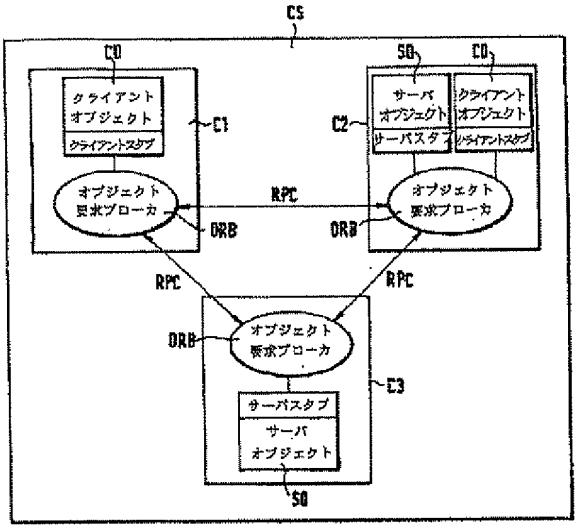


Fig.1a

【図１ｂ】

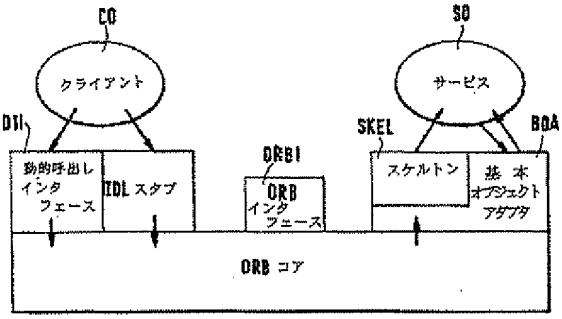


Fig.1b

【図２】

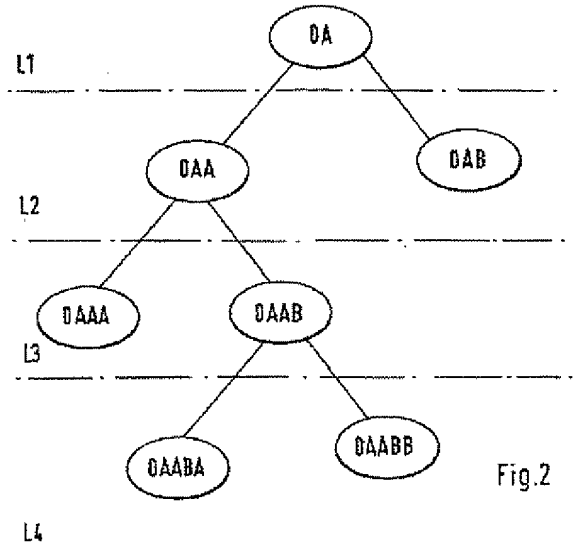


Fig.2

【図４ａ】

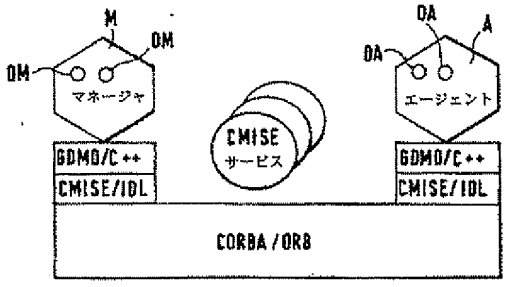


Fig.4a

【図3】

```
interface ManagedObjectCMISE {
...
      boolean Supports_subordinate( in ASN1_ObjectIdentifier oid );

      ObjectInstanceType Create_subordinate(
                              in ASN1_ObjectIdentifier oid,
                              in RelativeDistinguishedNameType name,
                              in Criteria criter
                              )
                  raises (NoFactory, InvalidCriteria, CannotMeetCriteria,
                          AutoNaming);

      ObjectInstanceType Delete_subordinate(
                              in RelativeDistinguishedNameType name,
                              )
                  raises (NotRemovable, InvalidName);
...
}
```

Fig.3

【図4b】

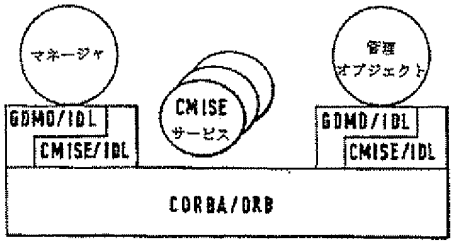


Fig.4b

フロントページの続き

(72)発明者 オリビエ・ポトニエ
フランス国、エフー75011・パリ、リユ・
ドユ・フオブール・サンータントワーヌ、
9

(54)【発明の名称】 第1のオブジェクトの生成をサポートするための方法、第1のオブジェクトを生成するための方法、第1のオブジェクトの削除のための方法、プログラムモジュール、およびコンピュータユニット

[外国語明細書]

1. Title of Invention

Procedure for the support of the production of a first object, procedure for the production of a first object, procedure for the deletion of a first object, programme module and computer unit

2. Claims

1. Procedure for the support of the production of a first object in an object environment in which the first and other objects interact through the CORBA-mechanisms, such that by means of the procedure a second object, which is a CORBA-generating object comprising at least one function for the production of the first object, is made available c h a r a c t e r i s e d i n t h a t the first object which is assigned as a second object to a directly preceding parent-object in accordance with a containment tree and that at least one function for the production of the first object corresponding to a CORBA-generating object is integrated into this parent-object.

2. Procedure in accordance with Claim 1 characterised in that the assignment is registered in a central service.

3. Procedure in accordance with Claim 1, characterised in that at least one CORBA-generating-object corresponding to a function for the deletion of the first object is integrated into the parent-object.

4. Procedure in accordance with Claim 1, characterised in that CMISE-functions are integrated into the parent-object as production functions.

5. Procedure in accordance with Claim 1, characterised in that the production functions of the parent-object are passed on to a CORBA child-object.

6. Procedure for the production of a first object within an object environment in which the first and other objects interact through CORBA-mechanisms such that in the course

of the procedure a production-message is sent to a CORBA-generating object, c h a r a c t e r i s e d i n t h a t the production-message to the first object is sent to a directly-preceding parent-object in accordance with a containment tree and that the parent-object functioning as a CORBA-generating-object, carries out the production of the first object.

7. Procedure for the deletion of a first object within an object-environment, in which the first and further objects interact by means of CORBA-mechanisms, such that as a result of the procedure, a deletion-message is forwarded to a CORBA-generating object, c h a r a c t e r i s e d i n t h a t the deletion-message to the first object is sent to a directly-preceding parent-object in accordance with a containment tree and that the parent-object functioning as a CORBA-generating-object carries out the deletion of the first object.

8. Programme-module with a CORBA-interface for interaction as a CORBA-object by means of CORBA-mechanisms and with a set of primary functions for the establishment of application-services, c h a r a c t e r i s e d i n t h a t the programme-module contains one or more secondary functions which have such a form that they make available a service as a CORBA-genereting-object, which carries out from the programme-module the production of child-objects which from a logical point of view with respect to a containment tree are directly-succeeding child-objects.

9. Programme-module in accordance with Claim 8, characterised in that the first functions are application services for the network management.

10. Programme-module in accordance with Claim 8, characterised in that the semantic of the second function is a CORBA-generating-object.

11. Computer unit which operates on a programme-module in accordance with Claim 8.

12. Computer unit in accordance with Claim 11, characterised in that the computer unit is a network management component.

3. Detailed Description of Invention

The invention relates to a procedure for the support of the production of a first object within an object environment, in which the first and other objects interact by means of CORBA-mechanisms in accordance with the generic term of Claim 1, procedures for the production and the deletion of a first object in accordance with the generic term of Claim 6 or 7, a programme-module with a CORBA-interface for interaction as a CORBA-object by means of CORBA-mechanisms in accordance with the generic term of Claim 8 and a computer unit in accordance with the generic term of Claim 11.

To an increasing extent, object-oriented modelling is used as the architectural principle for the design of software to be used with divided computer systems. The CORBA-software (CORBA = common object request broker architecture) has such an architecture and is an important component of the OSA-Architecture (OSA + object-service architecture) specified by the Object Management Group (OMG).

The invention is based upon the procedure by which objects (managed objects) are normally produced and deleted in a computer system in accordance with the CORBA- Architecture. This is described by way of example in "Common Object Request Broker: Architecture and Specification r2.o", Object Management Group, Framingham, Massachusetts, 1995.

The production and deletion of objects is carried out by means of special objects, namely the generating-objects (factories). The sole function of such a generating-object is to carry out

the deletion and generation of objects for specific kinds of objects. A special service, the lifecycle service is used to define such generating-objects. A general generating-object is available. Proceeding from this, special generating-objects for every kind of object e.g. special classes of objects (object class) can be defined which permit the input of special production parameters.

In order to be able now to produce or extinguish only one special object, first of all the appropriate generating-object for the production and deletion of this special object must be found. To achieve this, a search message is sent to a special service, the factory finder service which is a part of the lifecycle service and the search criteria (type, location, ...) for this search entered. If the appropriate generating object is found, a requirement message (invocation) is generated and the generating object sent, which thereupon produces or extinguishes an object corresponding to the parameters contained in the invocation.

The disadvantage of this procedure is that, normally, finding a suitable generating object is a time-consuming activity and results in a further increase in communication loading for the computer.

The basic task of the invention is now to reduce the loading of the system in a computer system provided with CORBA-architecture.

This task is resolved by a procedure in accordance with the principle of Claims 1, 6 and 7, by a programme-module in accordance with the principle of Claim 8 and by a computer unit in accordance with the principle of Claim 11.

In this respect the invention is based upon the concept that objects in a CORBA-environment are responsible for the production or deletion of the objects immediately following them (descendant objects) in the containment tree. In addition, they consequently take over the role of generating objects for their child objects.

As a consequence of this procedure which differs from the CORBA-philosophy, the following advantages are obtained:

The number of objects in the computer system is reduced as a result, there is also a reduction in the system loading and the number of references.

Furthermore, there is no longer any requirement for the factory finder service and the computer load represented by the search procedure. They are easy to find because of their arrangement in the containment tree.

In addition, the system-consistency is improved. A succeeding object is produced only if its predecessor exists in the containment tree.

Further advantages result from hybrid systems in the general area of network management, in which a network-management system based upon CORBA exists. For one thing, the CMISE Services can be used for the production- and extinguishing functions, which are inherited from a simple interface in which all CMISE Services are held.

In what follows below, the invention is further explained in three detailed examples supplemented by accompanying drawings.

In a first explanatory example, the execution of the procedure in a computer system in accordance with the invention is described, which consists of one or more computer units in accordance with the invention and in each of which a programme-module in accordance with the invention is running.

Fig. 1 shows a CS Computer System with three computer units, C1 to C3 which communicate with one another.

Typically, the computer units C1 to C3 consist of computers, printers or network elements of a communication network. Each has a hardware-platform consisting of processors, memory facilities and peripheral components, a software platform which includes, for example, an operating system and a databank system, and applications which are made up of application-programme-modules running from the software platform. The computer units C1 to C3 are connected together by one or more communication networks, for example, by X.25, #7, Ethernet or To-Token-Ring communication systems. The software-platform of the computer units C1 to C3 hereby provide the necessary data transmission services.

The application-programme-modules are modelled as objects (managed objects), i.e. the code and the data of an object are represented by a sum of attributes and functions to which other objects can have access. The application functions of the computer system CS result from the reciprocal access between a multiplicity of such objects.

In accordance with the CORBA-architecture, the computer units C1 to C3 exhibit several objects CO and SO and several object-request-brokers ORB.

From a service point of view, each of the objects CO and SO can be regarded as an encapsulated unit, which offers one or more services, which could be required by a client. In this way, the objects CO request a service (client object) ; which is provided by the SO objects (server objects).

To request a service, the CO forwards a request to an SO. Such a requests contains the following information: one operation, one target-object, no or several parameters and, optionally, a request-context. After the service has been provided, the SO forwards an outcome message back to the CO which is specified for this request message.

In order that request and outcome messages can be sent and received, an interface unit IU is available to the SO and CO objects.

The object-request-brokers (ORB) make an infrastructure available which permits the objects to communicate in a divided environment. In this context, the location of the SO which is requested to provide the service is not important for the object CO, i.e. it is of no consequence in which of the other computer units it is to be found nor does it matter in which special platform or in which implementation procedure the object has been realised.

In this context, each object knows at least one object-request-broker (ORB) and also how it has to make contact with this local object-request-broker . Each object-request-broker knows how to make contact with other object-request-brokers and how it has to communicate with them. In this situation, he uses the RPC-procedure (RPC = remote procedure call mechanisms).With this, an object sends a request message and one of the object-request-brokers ORB ensures that the message is passed on to the target-objective by means of the CORBA-infrastructure built up by the object-request-brokers.

Fig. 1b represents an illustration of the communication mechanism for the communication between a CO and an SO. Fig. 1b shows a communication layer ORB core, a superimposed communication layer containing 5 function units DII, IDLSubs, ORBI, SKEL and BOA together with two objects CO and SO having access to these function units.

In order to be able to communicate over the CORBA-infrastructure and to be able to co-operate with other objects in this infra-structure, each of the objects CO and SO must have access to a CORBA-specific interface. In this context, such an interface contains a description of a set of possible operations which another object can require of this object. The object interfaces are defined here in the description language IDL (interface definition language) which consists purely of an interface description language. The inheritance of this interface permits that one object supports several interfaces.

In CORBA, access is made to an object directly through this CORBA-specific interface. The implementation of this interface is the object itself. It consists of the code and data and requires no agent entity as is the case if an object is represented purely by data structure.

In order to send a request message, the object CO needs to have access to the reference (object reference) of the object SO and needs to know about the type of the object SO and the operation which it is to carry out. The object CO initiates the request message by calling up sub routines of the function unit IDLStube or by producing the request message dynamically by means of the function unit DII (dynamic invocation interface). The second procedure makes it possible to request a service which was still unknown at the time the object CO was developed.

The receipt of the request message at object SO is supported by means of the function unit DOA (basic object adapter). It is also possible for the object to offer an interface by means of the functions of the function unit SKEL, which corresponds to the above-mentioned second opportunity.

There is a logical relationship between the objects of the computer system CS, the structure of which, by way of an example, is demonstrated in Fig. 2.

Fig. 2 shows seven objects OA to OAABB which have a relationship between them. The relationship of the objects to one another is known as the containment tree or containment hierarchy. Each object represents an example (object instance) of a special object-class. The object-classes have a special hierarchy, thus for example, a more specialised object-class is contained within more-generalised class and a more generalised object class contains several more specialised object-classes. The same applies to the concrete objects (managed objects, object instances). The object OA forms the root of the containment tree. The objects OAA and OAB are more specialised objects which are contained in the higher object OA. The objects OAAA and OAAB are objects which are contained within the object OAA and the objects OAABA and OAABB are objects which are contained in object OAAB The higher object is also described as the parent-object of the contained objects, which can yet again be described as their child-objects.

In CORBA, the generation and deletion of objects by means of special generating objects (factories) can be carried out in exactly the same way as the request for and provision of services. Each of these generating-objects contain a set of operations which are suitable for the production and deletion of any special kinds of objects at any one time.

This set of operations is now transposed from the special generating-objects to other objects, the main task of which is really the provision of quite different services. The transposition of these operations proceeds in accordance with the following scheme: the set of operations is transposed into each parent-object which is responsible for the production and deletion of the assigned child-objects in accordance with the containment tree. By generation of a child-object to a parent-object this set of operations is inherited and can be further specialised in the generation process, so that more specialised

production functions, for example, with more parameters, become possible. Objects (managed objects) thus form generating objects (factories) for their direct successors in the dependency tree. To administer these production and deletion activities for objects, the object stores a list of the types of objects which can possibly succeed it directly.

In this way, in addition to the operations which were originally present, the interface of this object also contains the production- and deletion functions for the directly-succeeding objects in the containment tree.

If an object is to be generated or deleted in this way, a corresponding request message is sent to the appropriate parent-object. The semantics of this request message is that of a generating-object. Because of the logical relationship, it is easy to localise the parent-object so that it is simple to find the responsible generating-object.

The possibility also exist that this form of producing and extinguishing objects need not be applied to all the objects of the computer system CS but rather only to the objects assigned to a branch of the containment tree. It would even be possible to separate the deletion- from the production function. Consequently, the deletion operations could also be integrated into the interface of the of the object to be deleted.

If it is still necessary first to specify generating-objects for special classes of objects, it is also possible to reach back to existing functions already present.

If CMISE-operations are available in the computer system, then CMISE-operations can be used for the production- and deletion functions and can be further inherited in a CMISE-interface. The relevant part of such a further inheritable interface is specifically represented in Fig. 3 in a description language.

In this connection, the operations in Fig. 3 have the following significance:

- Support_subordinate is in this connection a specification of the generic factory: : supports operation. It has an object-identifier as input instead of a more general key.

- Create_subordinate is a specialisation of generic factory:: create_object. Once again the object_identifier is used instead of a more general key. An extra parameter, "name" is present, which uses the attribute of the produced object to give it a name. However, the parameter "name" can also be contained within the last parameter "criteria". This is the same as in generic factory :: create_object one. We have a list of <name, value> pairs, which can contain any one of the production parameters (initial values, resource constraints, location,.....). The reference object, if one is present, is stored in a criterion. Exceptions are reported back by generic factory :: create_object. Except for a back report, autonaming, which indicates that the name parameter has been ignored, the objects are automatically named.

- Delete_subordinate is a specialisation of lifecycle object :: remove. This requirement is directed to the parent-object so that this can administer the resources which become free after the deletion of the object. This operation can itself release a reverse operation in the reversed object itself so that it can check whether or not it is capable of being reversed and thereby of extinguishing its own resources. Delete_subordinate can extinguish the 'not-removable variation', exactly as by means of lifecycle object :: remove. It possesses an additional invalid name variation which expresses the fact that the name made available does not identify any existing child object.

The second explanatory example provides a situation from the network management area.

In a second explanatory example, a description is given of the performance of the invention-based procedure in an invention-based computer unit which consists of one or more invention-based computer units, in each of which an invention-based programme-module is running. Contrary to the first explanatory example, this situation in respect of the computer system relates to a network management system in which non CORBA objects are so modified by adaptation procedures that from the outside, they behave on the CORBA infrastructure as if they were CORBA objects. The computer system is equipped like the computer system CS in Fig. 1a. The computer units represent network management units, for example, network elements, network management centre or a mediation device . It is also possible for the computer system to undertake still further tasks such as administration or provision of network services. These services could also be based upon the TINA-software architecture (TINA DPE Service Specification, TINA-C, 1994).

Fig. 4a is an illustration of the communication mechanism for the communication between two such network-management-objects through the CORBA-infrastructure.

Fig. 4a displays a communication layer CORB/ORB, several CMISE services which are generally available by means of this communication layer, two network-management-components M and A and at any one time two communication functions GMO/C++ and CMISE/IDL situated between the latter objects and the communication layer CORB/ORB.

Similarly, an object-model (management framework for open systems interconnection, ITU-T Recommendation X.700, 1992) has been standardised by the OSI (open system interconnection) for the area of network management. In addition to the object-model (SMI = structure of management information), basic objects, a set of management services (CMIP = common management informa-

information protocol) have also been specified for communication between the objects. Objects are specified in the description language GEMO, which uses the ASN Syntax and contains additional macros of its own.

In the case of the Components M and A, these are not CORBA-objects but rather one or more OSI-objects OM or OA and a manager or agent function unit. By means of the manager or agent function unit, operations are carried out on the objects or request messages are sent to other objects.

Agent and manager function unit communicate through the CMIP-protocol. From the point of view of the network management the component M takes on the role of a manager and component A that of an agent.

The communication unit GDMO/C++ consists of one or more special access-objects, which make possible the execution of CMISE operation upon the object OA or the object OM.

The CMISE management services are realised by a CMISE-object on the side of the object OA. The interface unit CMISE/IDL contains this CMISE-object and the services assigned to this object. The CMISE-object of the interface- CMISE/IDL is specified through an IDL-interface object and, externally, operates and appears like a CORBA-object. In order to render this specification possible and, thereby, to provide a CORBA-interface to the object OA, a type conversion from ASN.1 into IDL types is required. In this way, CMISE-services are available as a set of CORBA objects. By means of the CORBA- request fed through the CORBA-infrastructure, CMISE-operations can therefore be carried out on the object OA. The same applies to the object MO.

The part represented in Fig. 3 in the interface is now further inherited to permit the production or deletion of objects.

A second possibility of binding OSI-objects through a CORBA-infrastructure is displayed in Fig. 4b, which represents a third explanatory example.

Fig. 3b displays a communication layer CORB/ORB, several CMISE services which are generally available by means of this communication layer, the objects OM and OA and at any one time the communication functions GDMO/IDL and CMISE/IDL situated between the latter objects and the communication layer CORB/ORB.

By means of the interface unit GDMO/IDL, the specified OSI-objects of the components A and M in GDMO are translated into a specification as an IDL interface. Access to such a specified object can be made by means of classic CORBA-messages. In this way, each of the OSI-objects are transformed into a pure CORBA-object. Since the specifications in IDL and ASN.1 are of a different nature, (interface description <-> object specification) a complete translation is not possible and only a subset of CMISE-services are offered through the interface unit GDMO/IDL. This means that only a subset of CMISE-operations can be carried out on the transformed CORBA-objects.

To facilitate the deletion and production of objects by means of CMISE services, an additional interface unit CMISE/IDL is provided. The production function (m-create) present in CMISE is rendered accessible by means of a CORBA interface and an application of this function to the transformed CORBA-objects thus made possible.

To produce an object in component A, an object of the component M or a special CORBA-object of the computer system CS must be

addressed for the administration of the generating object responsible for production and deletion of the object. This generating object is the parent-object of the object to be produced, which is equally an object of component A. In order to be able to gain access to the production function, access is made here through the CMISE/IDL interface unit to the CMISE-production function, which makes the parent-object accessible through this interface. The production of this child-object is thereby carried out by means of a CORBA-message making use of the ~~CEMISE~~ CMISE semantic and with the support of CMISE-services by means of a transformed CORBA object. In a reciprocal manner, access is made to the corresponding CMISE service to extinguish objects.

4. Brief Description of Drawings

Fig.1a shows a block circuit diagram of a computer system for a first explanatory example.

Fig. 1b is a functional representation of the structure of the software of the computer system as in Fig 1.

Fig. 2 is a symbolic illustration of dependency relationships between objects.

Fig. 3 is an extract from an interface description in a representation in a symbolic description phase.

Fig. 4a is a functional representation of the software structure of a computer system for a second explanatory example.

Fig. 4b is a functional representation of the software structure of a computer system for a third explanatory example.

Fig. 1a

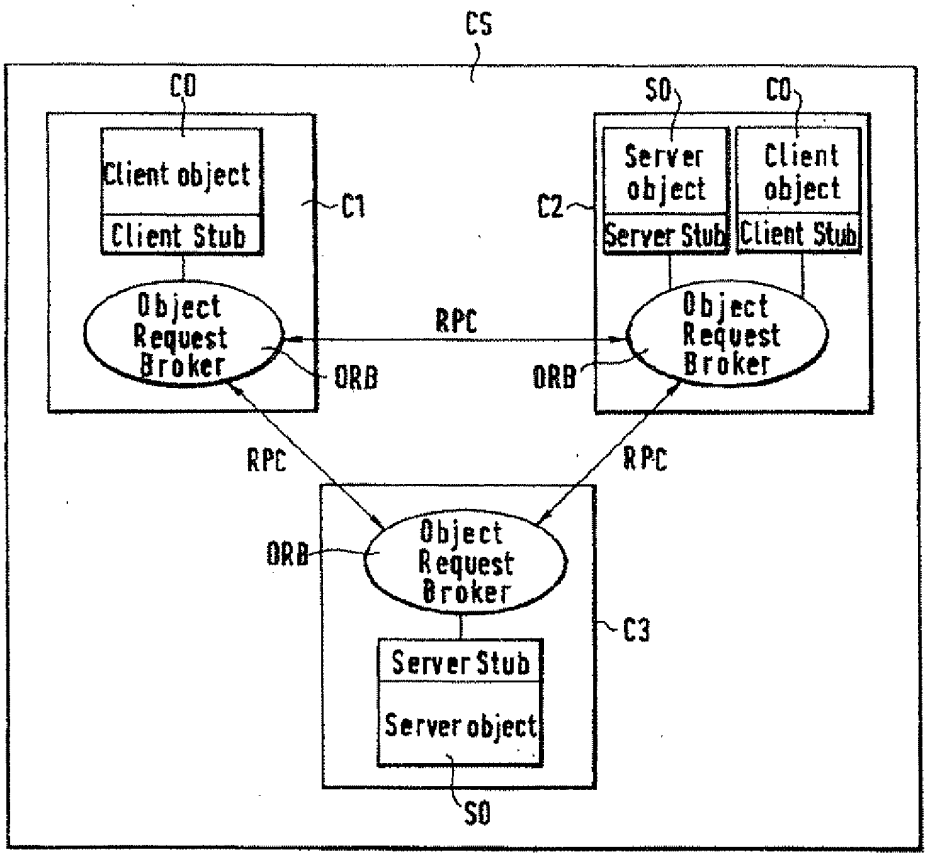

CS
CO
Client object
Client Stub
C1
Object Request Broker
ORB
RPC
SO
CO
Server object
Server Stub
Client object
Client Stub
C2
Object Request Broker
ORB
RPC
RPC
ORB
Object Request Broker
C3
Server Stub
Server object
SO

Fig.1a

Fig. 1b

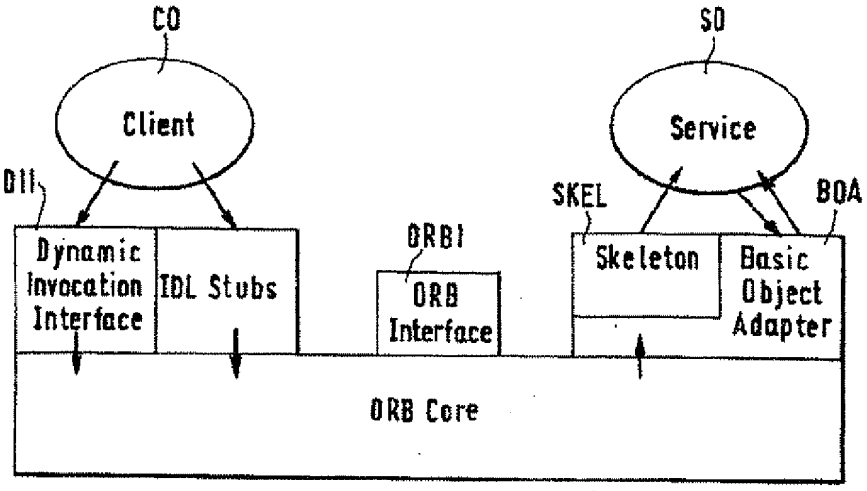

CO
Client
DII
Dynamic Invocation Interface
IDL Stubs
ORBI
ORB Interface
SO
Service
SKEL
Skeleton
BOA
Basic Object Adapter
ORB Core

Fig.1b

Fig. 2

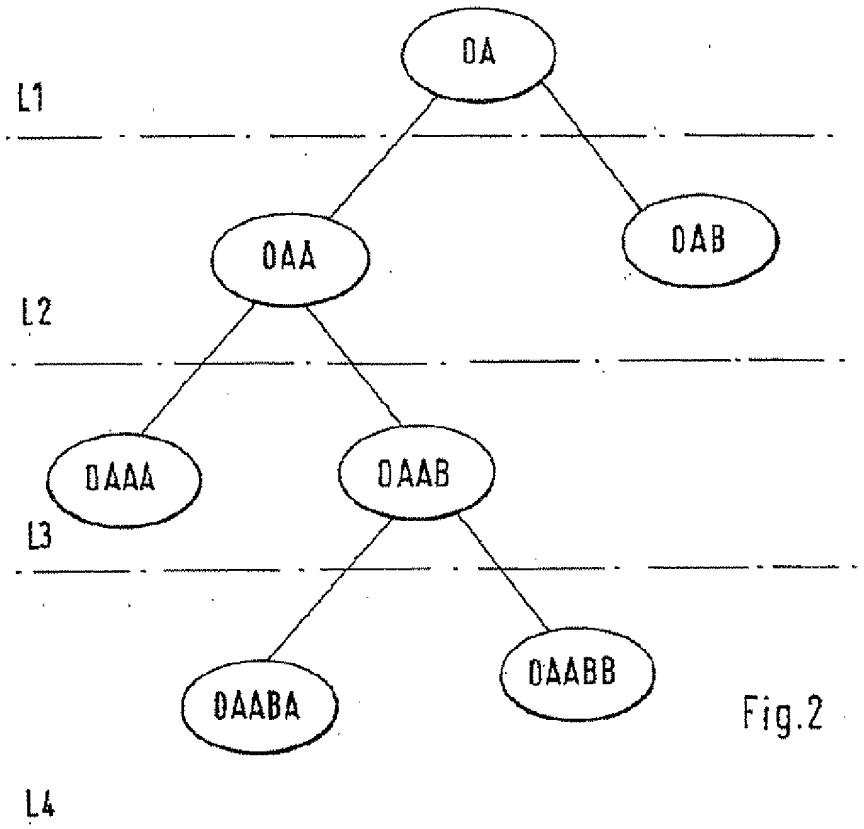


Fig.2

Fig. 3

```
interface ManagedObjectCMISE {
...
    boolean Supports_subordinate( in ASN1_ObjectIdentifier oid );

    ObjectInstanceType Create_subordinate(
                        in ASN1_ObjectIdentifier oid,
                        in RelativeDistinguishedNameType name,
                        in Criteria criter
                        )
                raises (NoFactory, InvalidCriteria, CannotMeetCriteria,
                        AutoNaming);

    ObjectInstanceType Delete_subordinate(
                        in RelativeDistinguishedNameType name,
                        )
                raises (NotRemovable, InvalidName);

...
}
```

Fig.3

Fig. 4a

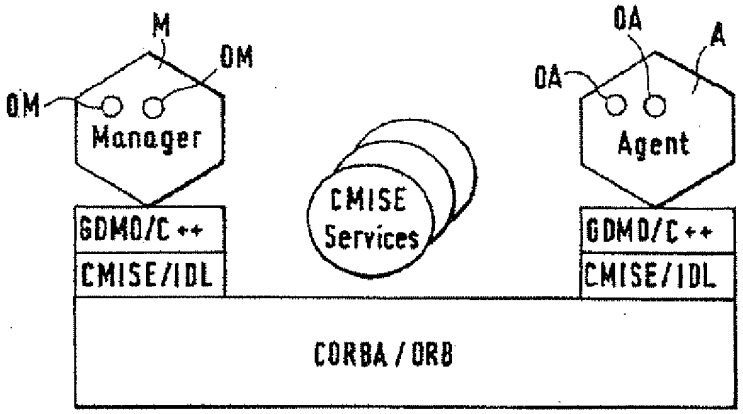


Fig.4a

Fig. 4b

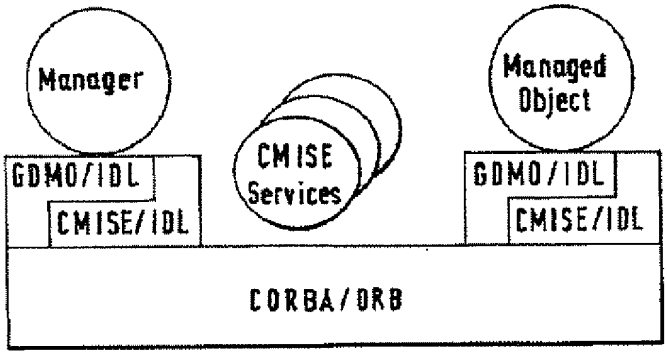


Fig.4b

1. Abstract

Procedure for the support of the production of a first object, procedure for the production of a first object, procedure for the deletion of a first object, programme module and computer unit.

First and other objects operate in an object environment by means of CORBA-mechanisms. A second object, which is a CORBA-generating object and which contains at least one function for the production of the first object is made available. A parent-object which is a directly-preceding object within a containment tree is now assigned to the first object, as a second object. Furthermore, at least one function for the production of the first object and corresponding to a CORBA-generating-object is integrated into this parent-object.

2. Representative Drawing

Fig. 2